Mechatronic Systems Engineering
School of Engineering Science
SIMON FRASER UNIVERSITY

MSE 312
TRUSS DESIGN PROJECT

Submitted by:	Group 13
Zheng Wang	301188444
Sohail Sangha	301186636
Rame Putris	301047157
Ane Tendo	301186203

Lab Demo Date:	June 11th 2015
Report Due Date:	June 23rd 2015

Contents

List of Figures

List of Tables

1 Introduction

The mechanical section of the Truss Arm project involves structure design, truss analysis and construction of a mechanical truss arm. The task for the arm is to pick up a 10g mass object from location A and transport the object along an arc to location B accurately and with fast speed. In general, the truss arm needs to have minimum deflection and minimum moment of inertia to have high operating efficiency and performance. After thorough truss analysis and simulation, the group designed and built an optimized truss arm to perform the task. This report briefly details the design process, decision, experimental and theoretical verification.

2 Design approach, objectives and constraints

With all the design specifications outlined, we start our truss arm design in SolidWorks, which allows us to run bending and deflection simulations easily. Equipped with this tool, we can compare alternative structures against parameters like bending and deflection during the design process in order to achieve an optimized design.

Given the brass rods supplied for cantilever beam our design objectives centered around achieving a design that: minimized beam deflection in the trusses as a measure to improve control accuracy during operation, and minimize the total moment of inertia to support rapid acceleration or deceleration.

With our objectives now clearly defined our design will have to span a length of 30 cm and support an additional suspended mass of 10g. Additional design specifications also set structure size limitations and maximum clearance space between electromagnet surface and object that we need to follow the table below outlines these

Parameter	Constraint
Truss Span	From center of motor mount to center of electromagnet mount is to be 30 cm
Height/Width	≤ 10 cm
Overhang (Motor/Electromagnet)	≤ 5 cm
Clearance	≤ 2mm
Truss Mounting Plate	The truss mounting plate is to be cut as shown below (dimensions are in mm)

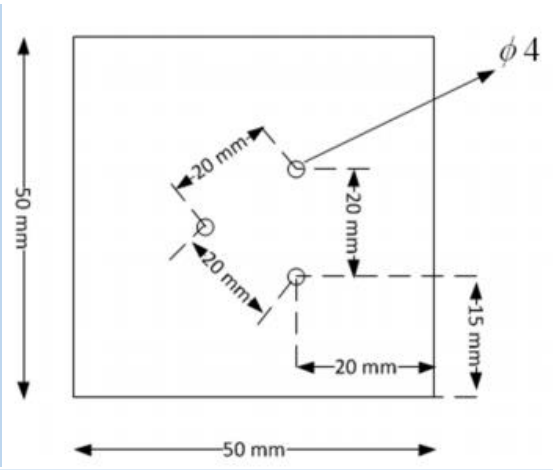
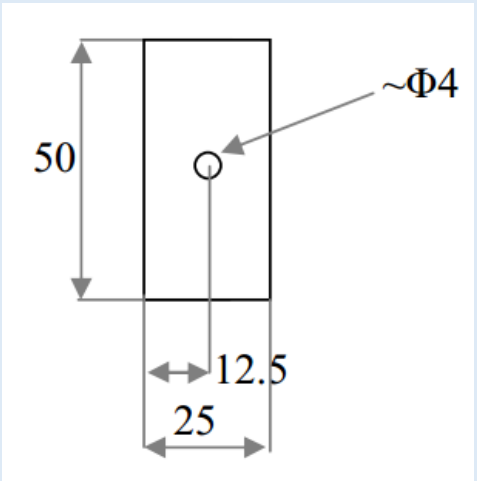
	
Electromagnet Mounting Plate	The electromagnet mounting plate is to be cut as shown in the figure below (dimensions are in mm) 
Brass Rod	12ft x 3/32” Rod

Table 1: Design Constraints of Truss

3 Quantitative analysis of 3 truss structures

3.1 Truss structure 1:

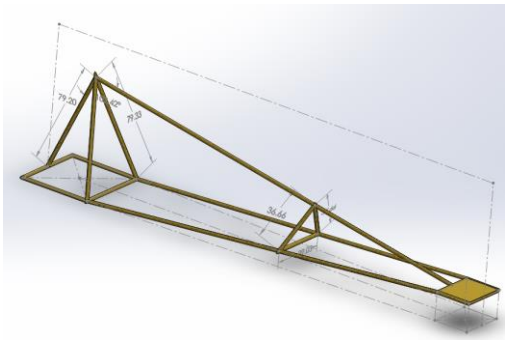


Figure 1: Truss structure 1 SolidWorks Model

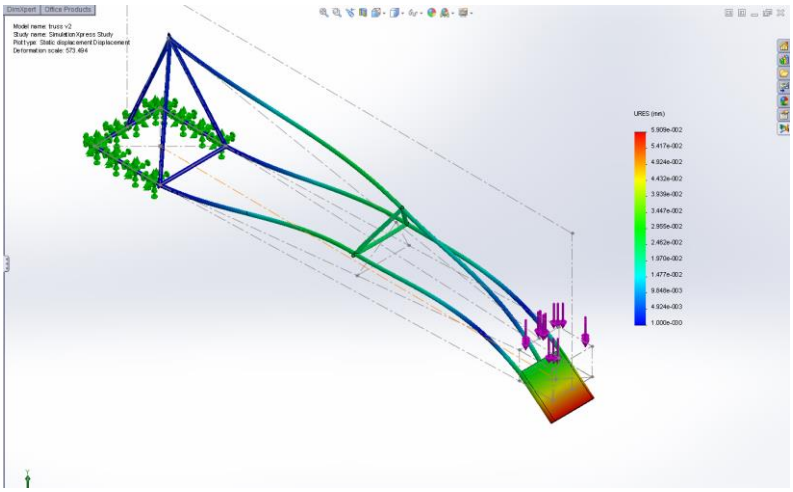


Figure 2: Truss structure 1 SolidWorks deflection analysis

Based on SolidWorks Mass Properties function, Principal axes of inertia and principal moments of inertia are (grams*square millimeters):

Iz = (0.00, 0.00, 1.00) Pz = 873749.23

3.2 Truss structure 2

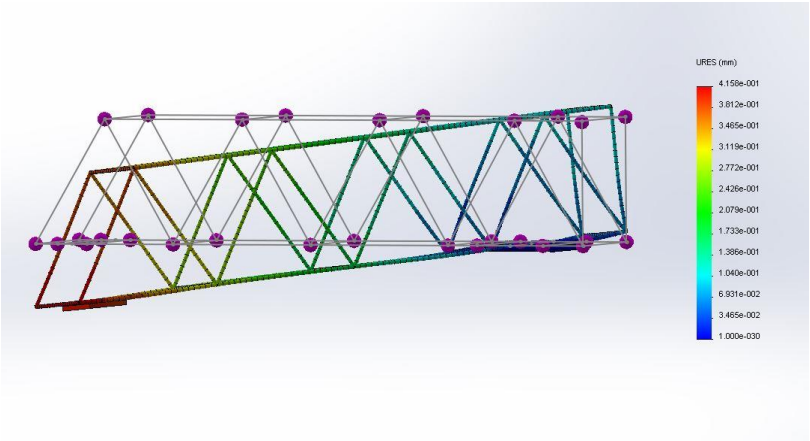


Figure 3: Truss structure 2 SolidWorks deflection analysis

Based on SolidWorks Mass Properties function, Principal axes of inertia and principal moments of inertia are (grams*square millimeters):

Iz = (0.00, 0.00, 1.00) Pz = 3206753.47

3.3 Truss structure 3

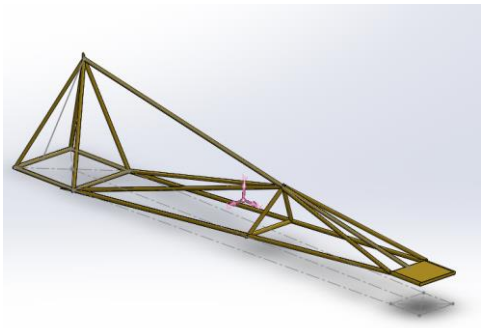


Figure 4: Truss structure 3 SolidWorks Model

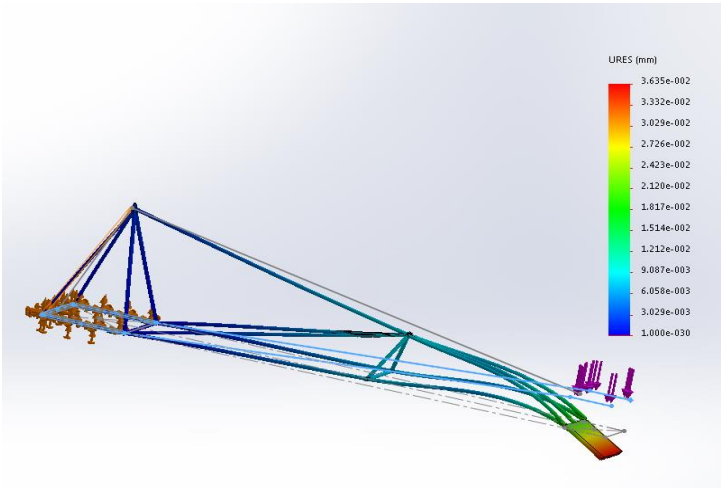


Figure 5: Truss structure 3 SolidWorks deflection analysis

Based on SolidWorks Mass Properties function, Principal axes of inertia and principal moments of inertia are (grams*square millimeters):

Iz = (0.00, 0.00, 1.00) Pz = 1030792.05

4 Design alternatives evaluation and justification

Given our principal concerns with every design is to minimize deflection in the direction of travel and as a result of the additional 10g mass picked up, and minimize the total moment of inertia of the structure for ease in control we compare these parameters for each design in the table below:

Design	z-axis deflection (x 10^{-2} mm)	Moment of inertia (g.mm^2)
Truss 1	5.909	873749.23
Truss 2	41.58	3206753.47
Truss 3	3.63	1030792.05

Table 2: Comparison of design options against system parameters of interest

From the table we found our design of choice was a debate between truss 1 with a lower moment of inertia but higher deflection with respect to truss 3. We favored Truss 1 over too as we value simplicity and reliability of our control algorithm and the 2mm difference in deflection is not too much of a problem.

5 Detailed Engineering drawing of final design structure

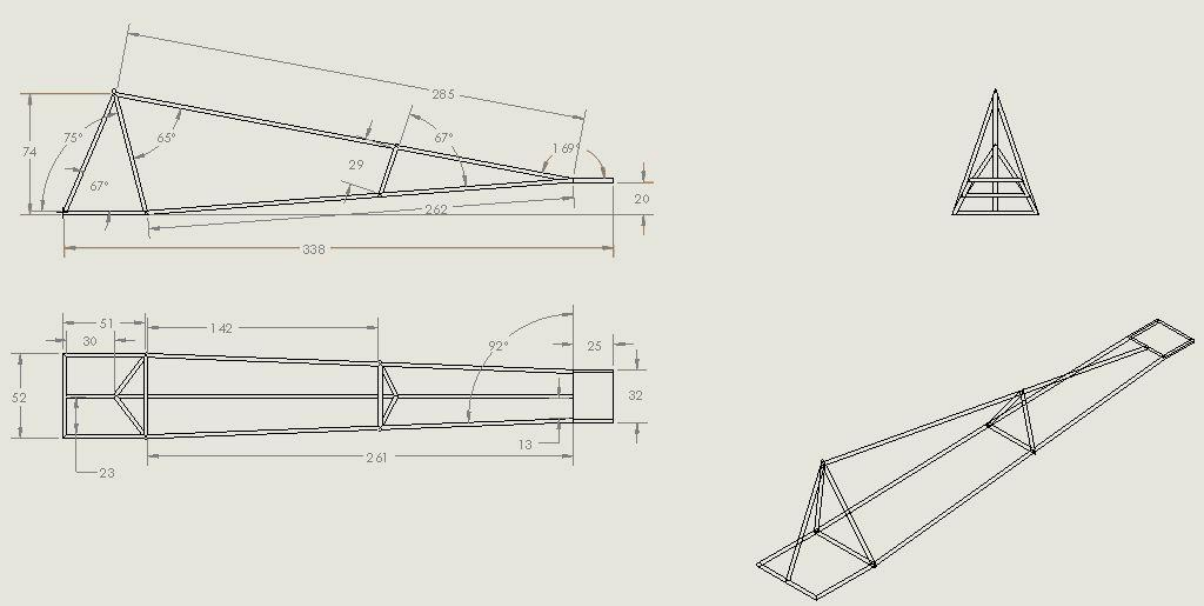


Figure 6: Detailed Engineering drawing of Design of Choice

6 Analytical analysis for structure stress and deflection

The analysis of the truss was not done explicitly by hand, but the equations were derived from scratch and fed into a Matlab script, thus reducing the tediousness of having to deal with 3 dimensional vectors. Following is a brief description of the structural methodology and the mathematical equations used to solve for parameters of interest.

In the real-life structure, steps were taken to reduce the complexity as well to increase the robustness by reducing the net number of joints. This was achieved by bending some of the elements in order to replace multiple independent members with a composite piece. For structural analysis purpose the mathematical analysis of the system will be too complex and redundant. To simplify the analysis we choose to go with the conservative method of replacing all joints (including bending) into rotatable ones. By taking this step we can insure that the final design will be by far stronger and reliable than the analytically design. Figure 7 is a diagrammatical representation of the simplified structure with details implying the constraint and loading positions.

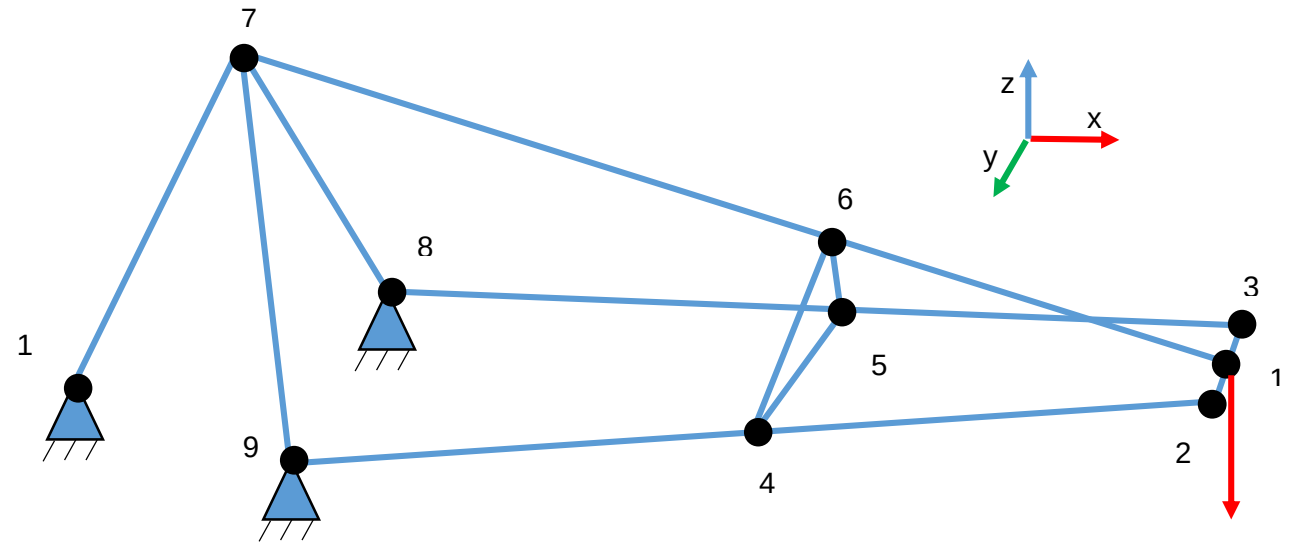


Figure 7: Numerical Assignment to joints

In the real structure, the three joints 8, 9, and 10 (as given in figure above) are attached to the square plate sitting on the motor thus are modeled as fully constrained. On the front end, the structure has been simplified by replacing the electromagnet plate with a simple member. As such joint 1 does not indicate two members and joint 2-3 is a single member with three members attached to it at joint 1, 2, and 3.

Following is a diagrammatical representation of the loadings experienced by member 2-3.

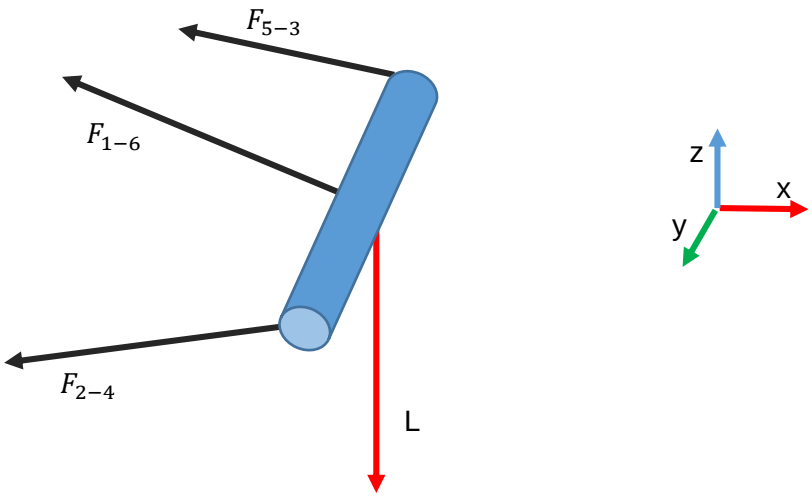


Figure 8: Member loading of structure

As shown in the figure above, solving the forces for joint 2-3 requires to find 3 unknowns which can be easily achieved by composing the 3-dimensional static equation along x, y, and z axis. The equations are composed in the following manner.

$$J_n = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} \quad \mathbf{1}$$

$Where: x_n, y_n, z_n define\ the\ location\ of\ the\ joint\ n\ in\ 3\ dimension$

$$\widetilde{F_{n-m}} = F_{n-m} * \frac{J_m - J_n}{|J_m - J_n|} = F_{n-m} * U_{m-n} \quad (2)$$

$Where: F_{n-m}\ is\ a\ scalar\ and\ is\ defined\ as\ tensile\ for\ all\ the\ members$
$U_{n-m}\ is\ the\ normalized\ direction\ vector\ given\ by\ the\ end\ points\ of\ member$

$For\ any\ given\ joint\ n\ with\ one\ known\ force\ [F_{n-k}]\ and\ three\ unkowns:$

$$\widetilde{F_{n-k}} + \widetilde{F_{n-m1}} + \widetilde{F_{n-m2}} + \widetilde{F_{n-m3}} = \begin{bmatrix}0\\0\\0\end{bmatrix} \quad (3)$$

$$\widetilde{F_{n-m1}} + \widetilde{F_{n-m2}} + \widetilde{F_{n-m3}} = -\widetilde{F_{n-k}} \quad (4)$$

$$[U_{n-m1}\ U_{n-m2}\ U_{n-m3}] * \begin{bmatrix}F_{n-m1}\\F_{n-m2}\\F_{n-m3}\end{bmatrix} = -F_{n-k} * U_{n-k} \quad (5)$$

$$A * x = b \quad (6)$$

$Where: A\ is\ a\ 3 \times 3\ matrix\ with\ U_{n-m}\ as\ column\ vetors$
$x\ is\ a\ colum\ vector\ containging\ the\ unkowns$
$b = -\widetilde{F_{n-k}}\ is\ the\ known\ solution\ to\ the\ equations$

$$\therefore x = A^{-1}b \quad (7)$$

The above set of equations were used on joints 4, 5, 6, and 7 to find the loadings in all members of the truss once $F_{1-6}, F_{2-4}, and\ F_{5-3}$ were known. This was possible due to the fact that given the design of the truss all these joints have exactly four members joining each other. Coincidently, the same approach was used on member 2-3, owing to the three unknowns and depiction of the loading as being delivered from a member aligned with the z axis. Further computation of the stress, force, and buckling criteria, net deflection and net inertia was a simple iterative task on the data computed for the loadings and thus easily automated in the Matlab script.

7 Buckling analysis

Buckling analysis was carried out by comparing the compressive force carried by the member against the minimum force required to cause buckling in the member. This task was carried out in matlab script as well.

$$F = \frac{\pi^2 EI}{(KL)^2} \quad (8)$$

$Where: F\ is\ the\ minimum\ force\ required\ for\ buckling$
$E, I, L\ are\ the\ eleastic\ coefficient, Area\ Moment\ of\ Inertia, and\ Length\ of\ member$
$K\ is\ the\ effective\ length\ factor, 1.0\ for\ both\ ends\ pinned$

Member of arm	Loading [N]	Stress [N/m^2]	Factor of safety [Stress/Y.S.]	Buckle Likeness [Yes|No]/ Safe Limit[N]
1-6	3.85	784850.82	305.39	NA
2-4	-1.90	-386593.78	NA	No/270.69
3-5	-1.90	-386593.78	NA	No/270.69
4-6	-0.02	-3986.03	NA	No/2764.29
5-6	-0.02	-3986.03	NA	No/2764.29
6-7	3.86	785790.94	305.03	NA
4-5	0.04	8616.06	27818.87	NA
4-9	-1.90	-388039.93	NA	No/182.96
5-8	-1.90	-388039.93	NA	No/182.96
7-8	-3.25	-662856.99	NA	No/604.40
7-9	-3.25	-662856.99	NA	No/604.40
7-10	5.73	1166644.48	205.45	NA

Table 3: Buckling analysis data from Matlab program

8 Moment of inertia

The moment of inertia for the truss was computed about the rotational axis adjunct with the motor. This task was automated in the matlab script as well by using the following formulation:

$$Borrwing\ notation\ of\ J_n\ from\ earlier\ text$$

$$L = \sqrt{J_m - J_n} \tag{9}$$

$Where: L\ is\ the\ length\ of\ the\ member\ m-n$

$$\begin{bmatrix} x_{m-n} \\ y_{m-n} \\ z_{m-n} \end{bmatrix} = \frac{J_m - J_n}{L} \tag{10}$$

$$\sin(\theta) = \frac{\sqrt{x_{m-n} + y_{m-n}}}{L} \tag{11}$$

$$r = \sqrt{(x_0 - x_{m-n})^2 + (y_0 - y_{m-n})^2} \tag{12}$$

$$I_{x_0 y_0} = \frac{1}{12}(AL\rho) \times L^2 \times \sin(\theta)^2 + (AL\rho) \times r^2 \tag{13}$$

$Where: x_0, y_0 are\ point\ in\ x-y\ plane\ coincedent\ with\ line\ of\ rotation$
$\rho\ is\ the\ density\ of\ member\ material$

Net I = 11562.830156 gm.cm^2

9 Appendix

9.1 Matlab Code for Analytical Calculation

```
function truss_solver()
    clc
    clear all;

    %setting up joint locations, (x, y, z)
    joints{1} = [31.25 0 1.99];
    joints{2} = [31.25 1.5 1.99];
    joints{3} = [31.25 -1.5 1.99];
    joints{4} = [19.405 2.05 1.158];
    joints{5} = [19.405 -2.05 1.158];
    joints{6} = [20.409 0 4.095];
    joints{7} = [3.092 0 7.307];
    joints{8} = [5.0 -2.5 0];
    joints{9} = [5.0 2.5 0];
    joints{10} = [0 0 0];

    %setting initial parameters
    Load = 1;                   %external load [N]
    Diameter = 0.25;            %beam diamter [cm]
    YieldStrength = 2.39689*10^8;   %N/m^2
    ElasticModulus = 1019710*9.91;  %N/cm^2
    Density = 8.5;              %gm/cm^2
    rotational_x = 2.5;         %cm
    rotational_y = 0;           %cm

    %auto configured vars
    Area = pi*(Diameter/2)^2;
    AreaMoment = (pi/2)*(Diameter/2)^4;

    %solving front side
    A = getA(1, 6, 2, 4, 3, 5);
    b = -[0; 0; -Load];
    output = A\b;
    field1 = 'val';
    field2 = 'name';
    forces(1) = struct(field1, output(1), field2, '1-6');
    forces(end + 1) = struct(field1, output(2), field2, '2-4');
    forces(end + 1) = struct(field1, output(3), field2, '3-5');

    %solving joint 6
    output = solveJoint(forces(1).val, 6, 1, 4, 5, 7);
    forces(end + 1) = struct(field1, output(1), field2, '4-6');
    forces(end + 1) = struct(field1, output(2), field2, '5-6');
    forces(end + 1) = struct(field1, output(3), field2, '6-7');

    %solving joint 4
    output = solveJoint(forces(2).val, 4, 2, 5, 6, 9);
    forces(end + 1) = struct(field1, output(1), field2, '4-5');
    forces(end + 1) = struct(field1, output(3), field2, '4-9');

    %solving joint 5
    output = solveJoint(forces(3).val, 5, 3, 4, 6, 8);
    forces(end + 1) = struct(field1, output(3), field2, '5-8');

    %solving joint 7
    output = solveJoint(forces(6).val, 7, 6, 8, 9, 10);
```

```
  forces(end + 1) = struct(field1, output(1), field2, '7-8');
  forces(end + 1) = struct(field1, output(2), field2, '7-9');
  forces(end + 1) = struct(field1, output(3), field2, '7-10');

  %running loop on all joints to calculate paramters of interest
  netI = 0;
  deflection = 0;
  buffer = '';
  numMembers = size(forces);
  for i = 1:numMembers(2)
    jointNums = textscan(forces(i).name, '%d-%d');
    mLength = norm(joints{jointNums{1}} - joints{jointNums{2}});

    stress = forces(i).val*10000/Area; %calculate stress in member
    fos = sprintf('%.2f', YieldStrength/abs(stress)); %calculate factor of safety
    buckle = 'No';
    F_max = 0;

    if(forces(i).val < 0) %the force is compressive, check for buckling
      fos = 'NA';
      F_max = ((pi^2) * ElasticModulus * AreaMoment)/(mLength)^2;
      if(abs(forces(i).val) > F_max)
        buckle = 'Yes';
      end
    end
    buffer = sprintf('M: % 4s F: % 4.2f N stress: % 11.2f N/m^2 fos: % 9s buckle: % 9.2f/%s', forces(i).name, forces(i).val, stress, fos, F_max,
buckle);
    disp(buffer);

    %calculate inertia of member
    netI = netI + getI(jointNums{1}, jointNums{2}, rotational_x, rotational_y);

    %addition for net deflection
    deflection  = deflection + (forces(i).val^2)*mLength/(Area*ElasticModulus*Load);
  end
  disp('-------');

  buffer = sprintf('Net I = %f gm.cm^2', netI);
  disp(buffer);

  disp('-------');

  buffer = sprintf('Net deflection = %f cm', deflection);
  disp(buffer);

  disp('-------');

  disp('Done');

  %----- FUNCTIONS -----%

  %grab the component of member along certain axis
  %start - starting joint number of member
  %final - ending joint number of member
  %cNum - 1=x 2=y 3=z
  function [comp] = getComp(start, final, cNum)
    vec = joints{final} - joints{start};
    length = norm(vec);
    comp = vec(cNum)/length;
  end

  %A.F = b
```

```
  %get the linear equation matrix (A) for joints:1,2,3
  %j<n>_s - starting joint number of member <n>
  %j<n>_s - ending joint number of member <n>
  function [A] = getA(j1_s, j1_e, j2_s, j2_e, j3_s, j3_e)
    A = [getComp(j1_s, j1_e, 1), getComp(j2_s, j2_e, 1), getComp(j3_s, j3_e, 1);
    getComp(j1_s, j1_e, 2), getComp(j2_s, j2_e, 2), getComp(j3_s, j3_e, 2);
    getComp(j1_s, j1_e, 3), getComp(j2_s, j2_e, 3), getComp(j3_s, j3_e, 3);];
  end

  %get the output vector (b) for linear equations above
  %j1_s - starting joint number for external force direction
  %j1_s - ending joint number for external force direction
  function [b] = getb(j1_s, j1_e)
    b = [getComp(j1_s, j1_e, 1); getComp(j1_s, j1_e, 2); getComp(j1_s, j1_e, 3)];
  end

  %given a member with known force, solve the forces in other members
  %sharing a joint
  %knownF - force in the member
  %jk_s - starting joint for known member/shared joint between members
  %jk_e - ending joint for known member
  %j<n>_e - ending joint for unknown member <n>
  function [outputxyz] = solveJoint(knownF, jk_s, jk_e, j1_e, j2_e, j3_e)
    xyzA = getA(jk_s, j1_e, jk_s, j2_e, jk_s, j3_e);
    xyzb = -knownF.*getb(jk_s, jk_e);
    outputxyz = xyzA\xyzb;
  end

  %get Inertia of member about z axis
  %j_s - starting joint
  %j_e - ending joint
  %x,y - Inertia about point in x,y
  function [netI] = getI(j_s, j_e, x, y)
    vec = joints{j_s} - joints{j_e};
    length = norm(vec);
    comp = sqrt(vec(1)^2 + vec(2)^2)/length;
    mass = Area*length*Density;
    I = mass*(length^2)*(comp^2)/12;
    cg = (joints{j_s} + joints{j_e})./2;
    r = sqrt((cg(1) - x)^2 + (cg(2) - y)^2);
    netI = mass*r^2 + I;
  end
end
```